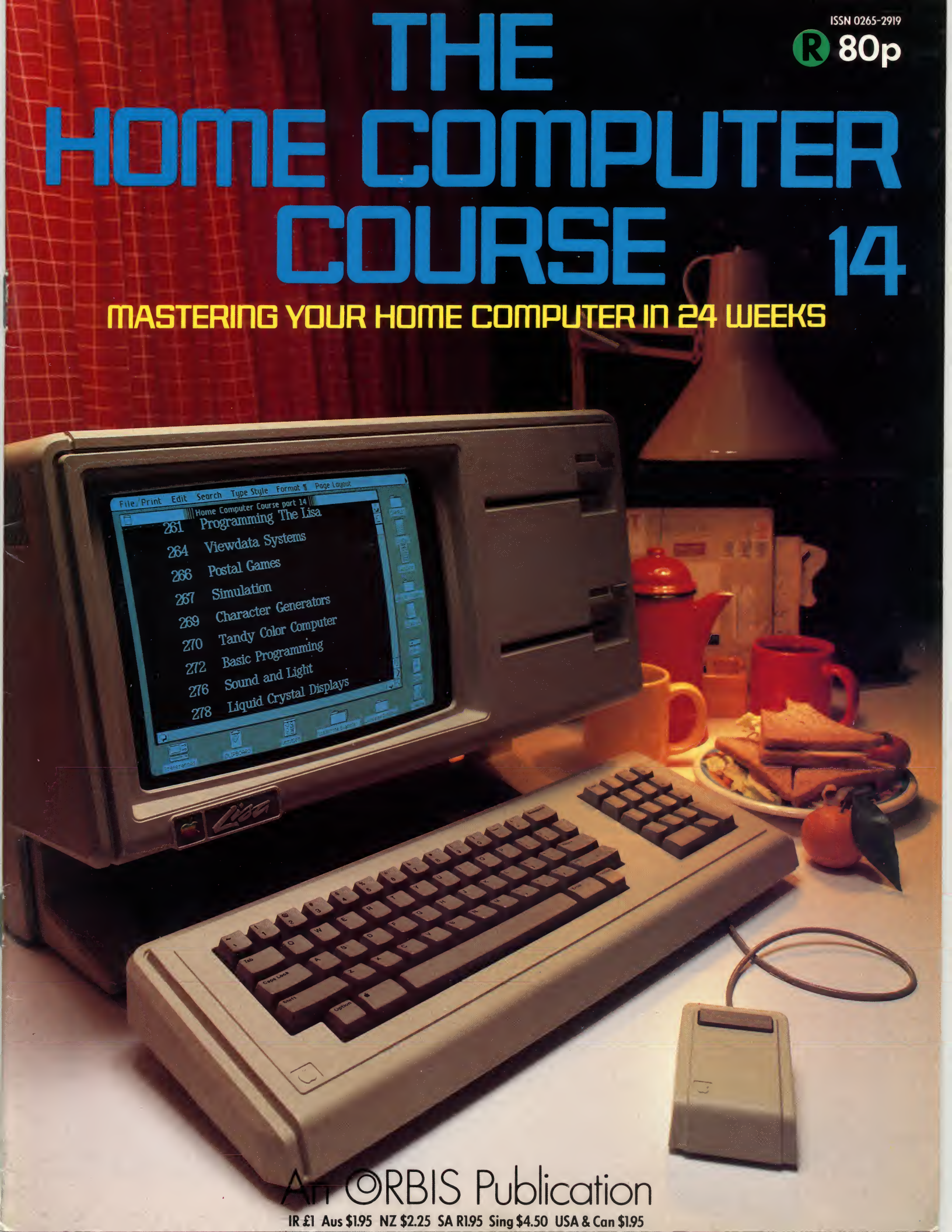
ISSN 0265-2919

80p

THE HOME COMPUTER COURSE 14

MASTERING YOUR HOME COMPUTER IN 24 WEEKS



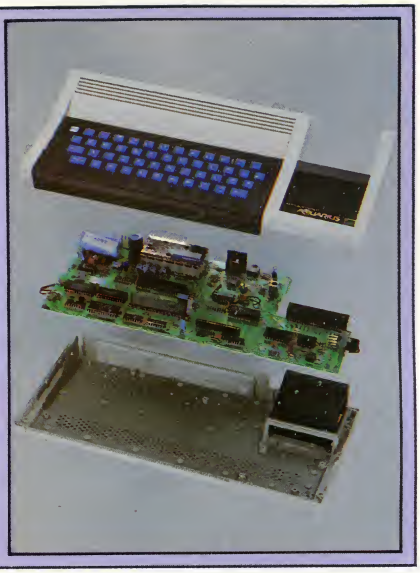
An ORBIS Publication

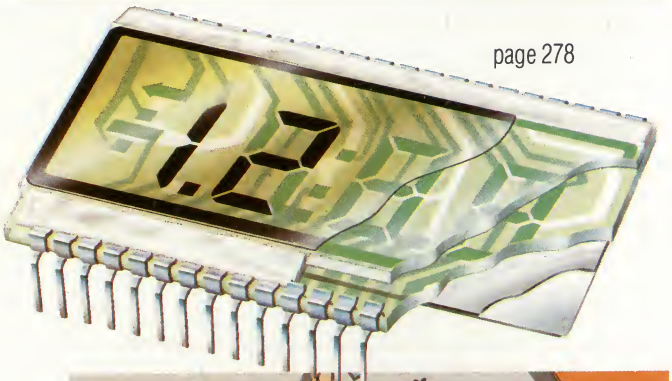
IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

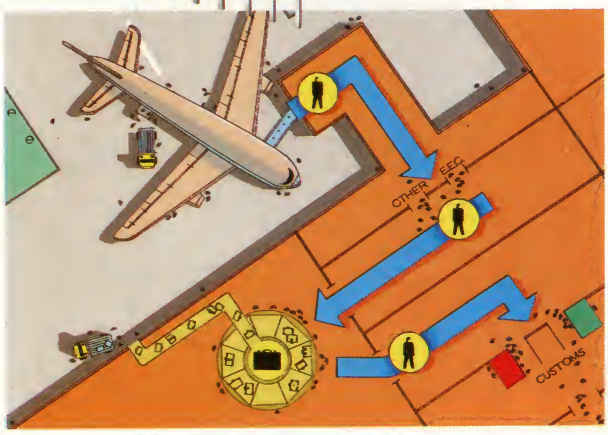
Next Week

- We turn our attention to robots: how they are controlled by computers, and what the small domestic devices you can buy have in common with industrial units
- We examine the Aquarius, a home computer that has clearly been designed with games playing in mind, but which is well supported with add-on devices
- Mazes are a common feature in computer games — both flat and three-dimensional. We show how they are programmed





page 278



page 268

Editor Richard Pawson; **Consultant Editor** Gareth Jefferson; **Art Director** David Whelan; **Production Editor** Catherine Cardwell; **Staff Writer** Roger Ford; **Picture Editor** Claudia Zeff; **Designer** Hazel Bennington; **Art Assistants** Liz Dixon, Safu Maria Gilbert; **Sub Editors** Robert Pickering, Keith Parish; **Researcher** Melanie Davis; **Contributors** Tim Heath, Henry Budgett, Brian Morris, Elizabeth Coley, Richard King; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Consultant** David Tebbutt; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1;

HOME COMPUTER COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER COURSE – Copies are obtainable by placing a regular order at your newsagent.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 4, 5 and 6. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Miller (Malta) Ltd. M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595. Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

Picture Books

Apple's Lisa is the most innovative business computer on the market. Many of its features, however, will eventually appear on home computers

The Lisa, produced by Apple Computer, is a machine intended purely for business use that costs £6,500, not including the printer. So you might well ask what place such an expensive machine has in THE HOME COMPUTER COURSE. The reason we've chosen to give it so much attention is that the Lisa is very much ahead of its time, and many of its features will eventually be used in home computers. Apple are known to be working on scaled-down versions of the project already, and rival manufacturers are rushing to emulate its capabilities.

It is not the hardware that is so radically new about the Lisa, but rather the standard software that comes with it. Developing sophisticated software is, in fact, a general trend for all microcomputers at the moment. It now takes fewer man-hours to design and build a new type of microcomputer than it does to write a sophisticated arcade game or business program. Software is becoming the most important element in any computer system, and increasingly the most expensive as well. Home users now find that they can spend as much on cassette and cartridge programs in one year as they did on the machine itself.

Nevertheless, we'll discuss the Lisa's hardware first, the design of which was fundamentally dictated by the requirements of the software. The Lisa comes with one megabyte of RAM as standard (that is, one thousand times the standard memory on a ZX81). Such a large memory requires the microprocessor to spend quite a bit of its time in 'memory management' — moving data around in memory and keeping track of where everything is. The processor is a Motorola 68000, which is a 16-bit device (this means it is capable of processing 16 bits of data at a time, where most home computer CPUs handle eight). By home computer standards, it is a very fast processor with a very advanced instruction set. For permanent storage, the Lisa system includes two floppy disk drives, and a single hard disk drive — a detached unit with few external features. The hard disk is needed both for capacity (five megabytes) and speed — the Lisa makes use of a large number of programs that need to be exchanged frequently between the RAM and disk. We'll be looking at hard disks in more depth later on in the course, because low cost units are already appearing on the market for home computers.

Another striking feature of the Lisa's hardware is the built-in monochrome display, which has a resolution of 720 × 364 pixels. This permits a variety of different typefaces for text, as well as the kind of graphics shown in this feature. The Lisa's design incorporates special chips and circuitry solely dedicated to drive this display and rapidly move the images.

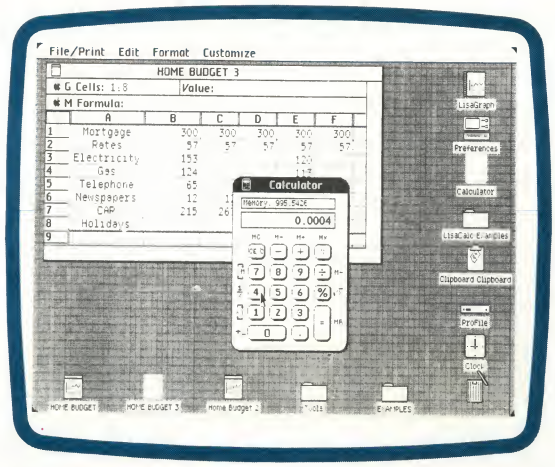
IAN McKINNELL

User Friendly
Apple's Lisa was designed to be used by people in a business environment who had no previous experience of computers. Use of a hand-held 'mouse' means that the keyboard is used far less frequently than on other systems

Linked to an appropriate printer — such as a high-speed, high-quality dot matrix unit — it is possible to reproduce on paper anything that can be displayed on the screen. However, if the printer is not compatible with the high level of screen resolution, the Lisa will produce a printed image of the highest quality possible.

The Lisa's keyboard is detachable from the main unit, and is well laid out. However, the keyboard is used less frequently than that of other machines, because the Lisa features a 'mouse'. A mouse is one of a number of alternative ways of inputting information to the screen without using the keyboard — other methods include the joystick, light pen and voice recognition units. The mouse is essentially a hand-held box, about the size of a packet of cigarettes, which is moved across

The Right Image
Every function that you perform on the Lisa is represented by a symbol called an 'icon'. To activate the function, the mouse is moved until the cursor is positioned over the icon, and the SELECT button on the mouse is pressed. This 'opens up' the application, which is displayed in detail on the screen



the surface of your desk or table, and is connected to the computer by a trailing wire. Moving the mouse causes a 'cursor' or pointer to move around the screen. In this way it is possible to point on the screen to the data or command you require, and then, by pressing the SELECT button on the top of the mouse, that data can be selected or that command executed. The keyboard needs to be used only when new data in the form of text or numbers has to be entered.

At this point, we are ready to discuss the Lisa's software, and again we stress that though the applications for Lisa currently fall entirely within the business area, the principles on which they operate will eventually filter through to home computer applications.

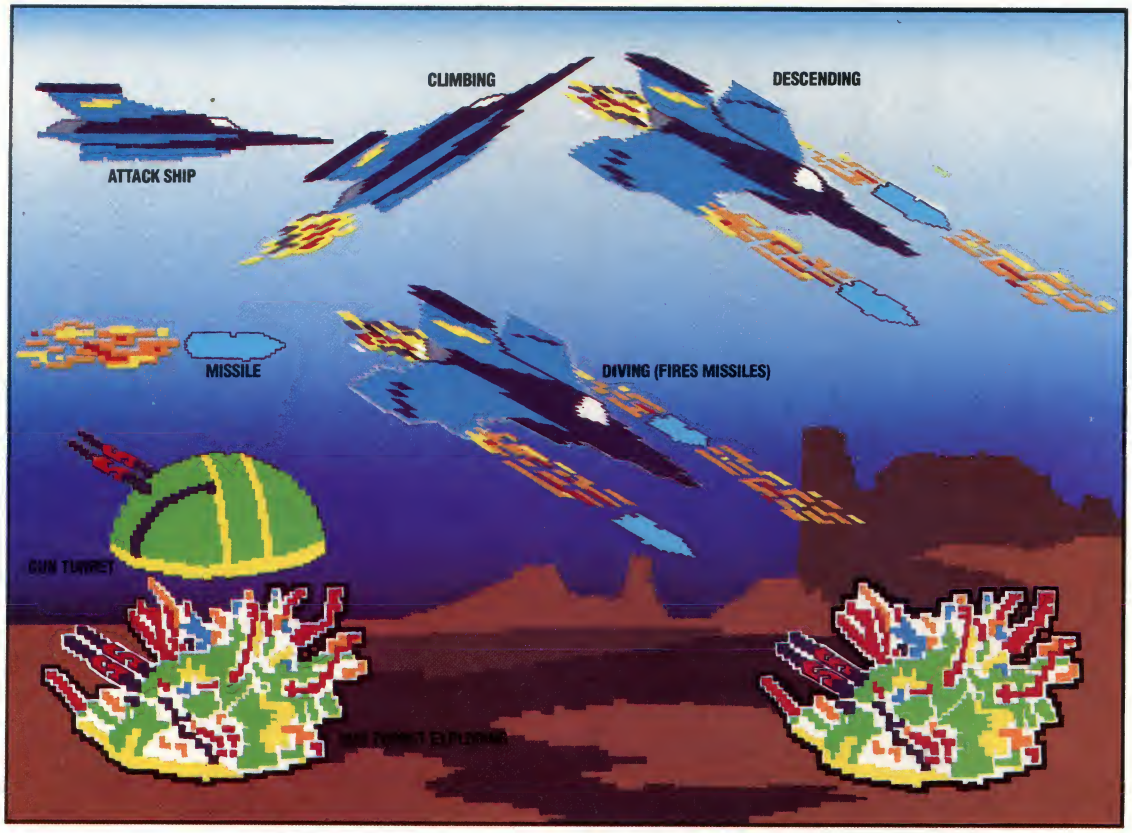
When you first switch the machine on, the image shown on the Lisa's screen represents a desktop with different items arranged on it. Indeed, almost everything that you do on the Lisa is represented as a visual equivalent of the way you would do something without the computer's help. This is the main reason why beginners find it so much easier to get to grips with the Lisa than with conventional hardware and software. The individual items arranged on the desktop are called 'icons' and each represents a particular function, which is usually labelled beneath it.

Take the example of the clock icon. Moving the cursor over the small clock using the mouse, and pressing the SELECT button will cause a much larger clock to be displayed on the screen, along with the date. If you don't want the large clock to clutter up the desk, it can simply be 'closed down' to its original size again. Similarly, selecting the calculator-shaped icon will 'open up' a larger calculator, which can be used for simple arithmetic. If you aren't happy with the layout of icons on the desktop, you can simply move them around the screen by holding down the SELECT button and moving the mouse. One of the most amusing touches, which highlights the degree to which the Lisa is modelled on the way we work, is the icon for the rubbish bin. If you no longer require any piece of work, you simply throw it into the rubbish bin icon using the mouse. Having this discipline means that it is very difficult to erase important information accidentally. It is even possible to examine the current contents of the rubbish bin and retrieve them, unless the Lisa has been switched off in the meantime!

Most real work is done on the Lisa using one of six applications systems: LisaWrite, a word processor; LisaCalc, a spreadsheet; LisaGraph, a

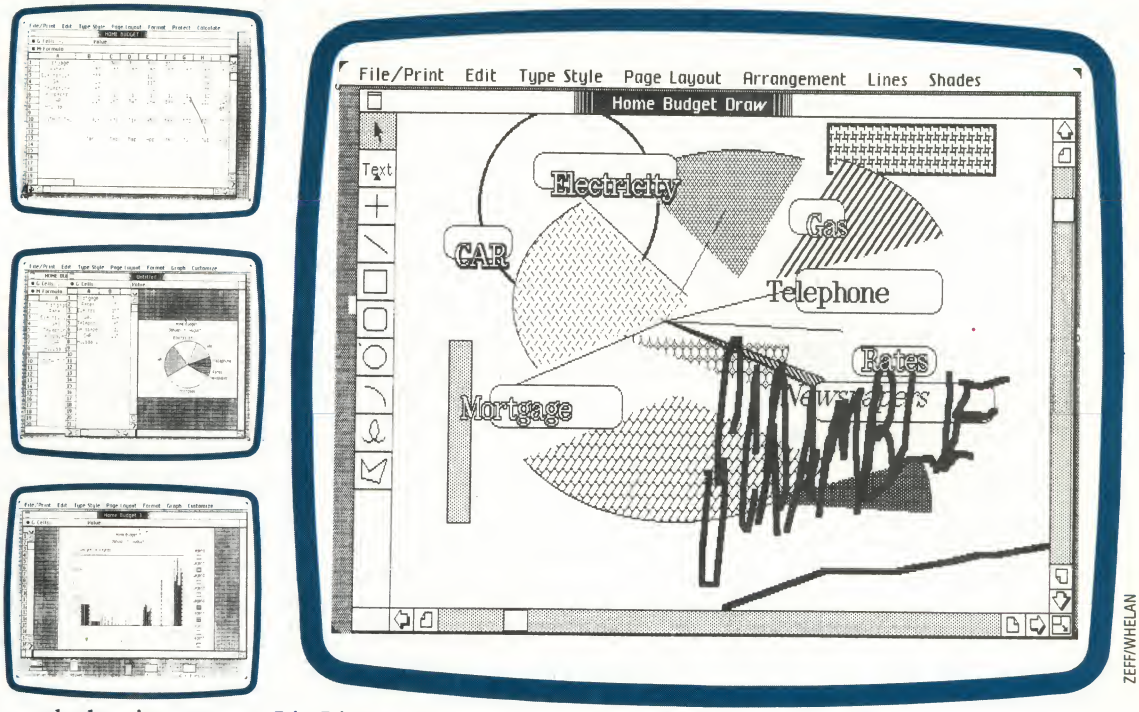
Object-Oriented Programming
To create a Defender-style arcade game using conventional programming would require you to design a number of sample screen layouts, and then write a program from scratch that controlled the whole game. Using object – oriented programming (which at the moment is difficult to do on home computers because suitable programming languages are not available), you would instead concentrate on each element in the game individually.

Starting with the attack ship, your definition would state that it always moved from left to right; that when the joystick is moved forward or back the ship moves up or down accordingly; and that when the FIRE button is pressed a missile is released. Your second definition concerns the missile. It defines its shape and states that it continues in one direction until it makes contact with another object, at which point it disappears. The static gun turret is defined by a simple shape, which changes to an image of an explosion if a missile comes into contact with it. Express these three definitions in a suitable language and string them together and the game is more or less written for you!



TONY LODGE

Passing It On
One of the Lisa's distinctive features allows for information to be passed from one application to another, using the COPY option. This stores information temporarily on the clipboard icon, ready to be 'pasted' into another window. For example, we might start off by analysing some figures using LisaCalc (the spreadsheet application). The resulting figures could then be copied onto LisaGraph, which would produce a pie chart or bar graph of the figures automatically. Finally, the entire image could be transferred to LisaDraw, which would allow us to embellish it with additional labels, arrows, diagrams or other details. The finished result can then be printed out



ZEFF/WHELAN

graph drawing system; LisaList, a database/list manager; LisaProject, a project planning aid; and LisaDraw, a sophisticated tool for creating all kinds of graphical images. The icons for these applications are simply pads of paper. To perform a spreadsheet calculation, for example, the cursor is positioned over the pad of paper ruled into rows and columns called LisaCalc. Pressing the SELECT button will then effectively 'tear off' a sheet of this paper, which is then placed elsewhere on the desktop and can be given a label such as 'sales projections'.

In fact, the user may choose to have several spreadsheet applications on his desk at a time, by re-selecting the same icon. In a normal computer system, you would have to go through the process of loading the spreadsheet program, and then specifying the data file that you wish to work on. On the Lisa, however, program and data are inseparable. This is another example of object-oriented programming, which we introduced when discussing the Pinball Construction Set (see page 241).

Another important feature of the Lisa 'operating environment' (as it is called) is its ability to 'window'. When an application is selected, it appears like a large sheet of paper on the desk. The size of this piece of paper can be specified, again by using the mouse. If there is more than one such application 'open' at once, the sheets will overlap each other, with the one you are currently working on displayed at the top of the pile — just as they would be on a desk. It may be that the application you are working on, for example the word processor, requires more space than the size of the sheet you've specified. In that case, the sheet merely acts as a window onto the application, and can be moved around to display any part of the whole document. The principle of windowing was fully explained when we first examined spreadsheets (see page 158).

It is quite possible to move information from one application to another, again by using the icons, and this is another major strength of the Lisa. Let's say you were doing an analysis of your monthly sales figures using LisaCalc. By using the COPY function, which is selected from a menu of special functions listed across the top of the screen, you could make a temporary copy of the results from the spreadsheet, which would be stored in the clipboard icon. Then, by selecting a piece of LisaGraph paper, those results could be entered into the INPUT DATA section of the graphing application, simply by selecting the PASTE option from the menu. If requested, LisaGraph will then produce a neat pie chart (or bar graph or line chart), fully labelled and shaded. Now, again using the COPY and PASTE options from the top of the screen, it is possible to copy this image onto a piece of LisaDraw paper. This last application would then allow you to embellish the pie chart with some arrows or diagrams, or change the labels and headings into a variety of type styles. The finished result could then be printed out and copied onto an overhead projector slide, or used as final artwork for a report or magazine article.

As we have said, the principles both of object oriented programming and Lisa-style operating environments will soon be filtering down even to the cheapest machines, particularly as they become more sophisticated in terms of processor speeds and RAM memory size. Imagine having multiple windows on your home computer's screen — you could write a program in one of them, and observe its output on another. Then you could call up programming aids simply by pointing to an icon shaped like a toolbox, and move subroutines around your listing simply by moving the mouse. Let's hope that such software capabilities aren't all that far away.

Windows On The World

Viewdata is one of the few areas of computing where international standards have been established. This gives your computer access to a network of databases across the globe

Viewdata systems, such as British Telecom's Prestel service, enable us to gain access to a variety of databases by way of a modified television set and a telephone. A system may be a public one, open and relevant to all, like Prestel; it may be open to all but rather esoteric in content — like Fintel, a constantly updated financial database; or it may be private.

Viewdata systems, however, should not be confused with the teletext systems now available over the broadcast television channels, which use similar character sets and page layouts. Teletext systems, such as the BBC's Ceefax or ITV's Oracle, are broadcast as subsidiary signals overlaid on normal television programmes. The signals that define a teletext page are transmitted between the frames of a normal television picture. A decoding device separates them out and creates the teletext image. The resulting text and graphics can displace the normal broadcast picture or be superimposed over it so that the two appear together, as when teletext is used to provide subtitles for the hard of hearing.

Teletext systems are not interactive — that is, there is no means by which the viewer can change the contents of a page of information, or even make a response. Access to them, however, is free. All that you require to enjoy the benefits of a large volume of useful information is a suitably adapted television set.

Viewdata systems, on the other hand, use the domestic television set simply as a monitor to display data received over the telephone network. Normally, users cannot amend the database but they can use their keypad to enter responses to questions into the system. All viewdata systems are totally menu-driven — that is, the range of choices available to the user at any level is always displayed. The user makes a choice by entering a number on the numeric keypad. This transfers control into the selected sub-menu; and the procedure is repeated down through the levels of the hierarchy until the user arrives at the desired page of information.

Each page is identified by a unique number, and it is possible to go straight to a particular page by entering that number. This is the quickest (and the cheapest) method for those people who consult a particular page frequently.

By way of an example, let us follow a hypothetical user as he books his summer holiday via Prestel. The 'sign-on' frame (the first picture he sees when he turns the system on) tells him to press the # symbol on the keypad to arrive at the main index. From then on he follows the menu prompts from page to page. First stop is the General Information frame. The fourth entry in this page is headed Holidays, Transport, Travel, so he keys 4.

In response he will be offered a choice between Rail Travel, Air Travel, Coach Travel, Other Transport, Holidays and Tourism, and Cars and Motoring. There are also four further 'exits', each of which goes to an information frame (a sort of detour, like a footnote in a book) from which he can return to the main body of the text. After pressing the key corresponding to Holidays and Tourism, our

COURTESY OF PRESTEL

Public Information
British Telecom's Prestel viewdata service, comprising some 250,000 pages of information, is available anywhere in the country — even from a public telephone under certain conditions. All that is required to access the database is a telephone line and a television receiver

COURTESY OF PRESTEL

On-Screen Statement
The Nottingham Building Society, in conjunction with the Bank of Scotland and British Telecom, have developed a system that allows subscribers to take care of many of their financial transactions directly from their own homes, using a Prestel adaptor supplied by the consortium. Customers can examine their bank and building society accounts, pay bills, communicate with other subscribers and even buy a range of goods and services

holidaymaker has to decide whether he will book travel and accommodation separately or opt for a package deal, and that decision will prompt his exit route from the page. As an alternative means of page selection, Prestel have also issued a Directory, from which the user can make immediate selection. For example, he can be referred directly to the page that lists a specific airline or hotel chain.

We chose travel as our example because it represents the largest single use of the entire system in the British Isles. Travel agents, in particular, are among Prestel's most enthusiastic supporters.

Conceived at British Telecom's research laboratories in 1971, the first viewdata system was in operation by 1976 in an in-house trial, and available to the public as Prestel in late 1979. British Telecom had originally called their first system Viewdata, but were forced to change to Prestel when it was ruled that viewdata was a generic term.

Viewdata is perhaps unique in that it became an accepted standard both nationally and internationally from its inception. A variety of computer manufacturers and telecommunication companies started producing systems of their own that use Prestel's protocols and data structures. The effect has been to create a world-wide network of local databases, all of which are accessible to any subscriber.

Viewdata systems require a telephone line to be open all the time they are in use and the user incurs this charge over and above any connection into the viewdata service. Additionally, there is the possibility of a third charge for the particular frame retrieved, but this is at the discretion of the information provider. There is also a small annual subscription. In an attempt to reduce users' telephone charges, Prestel have installed a number of 'local concentrators' — trunk telephone lines dedicated to the system — that connect a caller in, say, Glasgow with a number in London at local call rates.

The hardware required to use viewdata services is divisible into three main types. Most sophisticated, but also most expensive, is the purpose-built viewdata terminal. The majority of commercial users opt for this. The second alternative is to fit an adaptor to a domestic television set. There are a variety of such adaptors, ranging from a simple numeric keypad with manual telephone dialling to a typewriter-style keypad with fully automatic dialling. But at the heart of all of these devices is a dedicated microcomputer that decodes the incoming and outgoing signals. The third method, popular with home microcomputer users, is to buy viewdata software for a standard micro. This has become particularly attractive since the introduction of a service known as Micronet 800, which offers subscribers the ability to load computer programs directly over the telephone line. Furthermore, these adaptors will usually permit a Prestel page to be saved on disk, thereby eliminating the need for a constant open telephone connection.

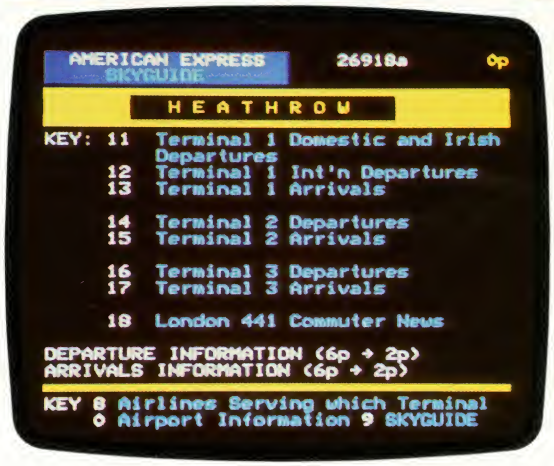
Prestel also enables subscribers to leave messages for each other via a service known as Mailbox. The subscriber is notified of a waiting message either as he switches on his Prestel terminal or, if his terminal is already in use, when he finishes a call. He can also check for messages at any time during a call. It is not necessary, as one might think, to have access to a full alphanumeric keyboard to send a message. There are a variety of standard 'message forms' that can be completed using numbers alone.

By mid-1983 there were some 35,000 Prestel subscribers in the United Kingdom, with more than a quarter of a million pages of information available to them, and an unknown number of companies and organisations using the viewdata specification for their own in-house database enquiries. This entire system and specification has been created in less than 15 years.

Distribution Network

The Micronet 800 service allows subscribers to acquire software directly from the central Prestel computer system. To use the system, it is necessary to receive the viewdata signals into a conventional microcomputer (not just an adapted domestic television), and then store the program either on cassette tape, or on disk, from which it can subsequently be loaded and run normally. Micronet offers around a hundred programs free of charge, with many more available at commercial rates. Micronet is available on most of the popular microcomputers, including the BBC Micro, some of the Commodore models, the Spectrum and the Apple II. Micronet subscribers also have automatic access to the rest of the Prestel database

COURTESY OF BRITISH TELECOM



Travelling Light
One of the most popular commercial applications for Prestel is in travel agents' offices. Airlines, in particular, have very sophisticated booking and ticket-issuing systems, though they all run on separate computers. Prestel allows agents access to most of these systems, and enables them to book tickets and reserve seats directly.

In addition, as shown in the picture on the left, it is possible to call up information about the arrival and departure of flights at United Kingdom airports. Users can also be warned about schedule changes

Winning Post

Not all computer games require split-second reaction times — postal games may take six weeks per move and involve several dozen players

Though most new purchasers of home computers declare that they intend to learn to program, there can be little doubt that the most popular application for such machines is actually playing games. As we have been at pains to point out in the HOME COMPUTER COURSE, computer-based games can be both entertaining and educational. There is no need to restrict yourself to arcade-style games such as Space Invaders, either. The computer has opened up whole new game concepts such as Adventure (see page 161) and educational simulations (see page 81), not to mention computerised versions of popular board games that offer a variety of different playing standards.

Nevertheless, there's one type of computer-based game we haven't mentioned before, which you may never have heard of. In marked contrast to arcade games, which require split-second timing and reflexes, these games proceed at a distinctly pedestrian pace, with each move taking weeks to play! And unlike most other computer games, which are for one player only, these games can involve several dozen players at a time.

We are referring to postal games — that is, games in which the players are scattered across the country (or in the case of international games, anywhere in the world), and in which each specifies his moves on paper at predetermined intervals. The moves are posted to a game co-ordinator, who feeds them into a microcomputer (most postal games don't require players to have a home computer of their own). The co-ordinator then sends a printout from the computer to each player, showing his own position and other relevant information such as the position of other players with whom he has made contact.

Such games usually continue for many months, if not indefinitely, though it is possible for players to join or leave the game at any stage. A joining fee is usually charged to cover the initial materials and rulebooks; thereafter a playing fee (typically around £1.00) is payable on each move. These games are by no means an inexpensive pastime! One move per week is considered to be an express game, while international games may involve a wait of six weeks between turns.

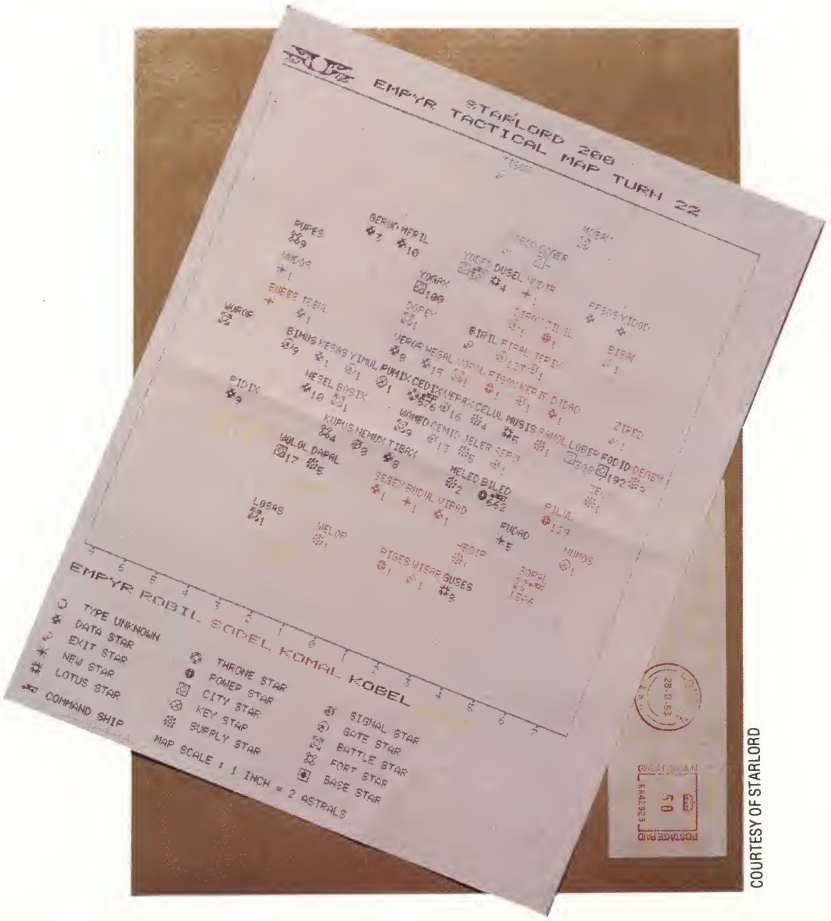
A typical postal game might cater for several dozen players. If more players wish to subscribe, the co-ordinator simply starts up a second game in parallel with the first, using the same programs but different data disks on his computer.

Postal games existed long before the computer. There were postal chess leagues, and the popular board game Diplomacy — in which players representing seven European nations aim to take over the continent by forming and breaking alliances with each other — can be played by post.

The introduction of the computer to do all the calculation and administration has meant that the scope of such games has increased, together with their ingenuity and sophistication. Some feature vast galaxies through which players manoeuvre their space fleets; others involve mythical lands and warring kingdoms. And there are computerised versions of Diplomacy, as well.

The unique thing about these games is that the players interact with each other — they aren't merely exploring, as with Adventure games. Alliances between ten or more players are not uncommon, which involves a lot of correspondence or telephoning each week. It is a measure of the quality of these games that the age range of subscribers is far wider than for most other types of computer game.

Galactic Wars
'Starlord' was the first postal game to be administered by computer in Britain. It is co-ordinated by Mike Singleton, using a Commodore Pet 3032 computer, a 7.5 megabyte hard disk unit, and an Integrex colour dot matrix printer. Each player's objective is to find the Throne Star and become Emperor. On each move he receives a map showing the immediate area around his forces, and a list of who controls nearby stars. The large disk capacity is dictated by the large number of programs that between them manage the game, and the large quantity of information that must be kept for over 700 players. For more information on postal games, a dedicated magazine called 'Flagship' is available from specialist game shops



COURTESY OF STARLORD

Model Behaviour

Simulation is a computer technique that allows you to experiment with what would otherwise be a dangerous or expensive situation. Home computer simulations can be very educational

One of the most important uses of computers is in the area of simulation. This is a method of forward planning in which a model of the situation to be analysed is 'simulated' on the computer. A model gives a simplified view of the situation, retaining the significant features of the problem and discarding minor details that will have little influence on the results.

Let us take the example of two glasses, one containing white wine and the other red wine. If a spoonful of the red wine is added to the white, thoroughly mixed and then a spoonful of the mixture is returned to the red glass, which glass has the greater impurity?

There are many ways to solve the problem, but one of the simplest is to use a model. For example, we could set up a model in which the volume of wine in each glass is exactly one spoonful. It is easy to see that in this instance both glasses will end up with the same degree of impurity (an equal mixture of red and white wine). Extending our model for larger quantities of wine will show that the same is true in these cases.

Models appear in three main forms. A pictorial model — for example, a photograph or a map — shows spatial arrangements and relationships between elements in that picture. Then there are models whose components behave in relation to each other in a similar way to the real elements they represent in the problem — for example, the problems solved by analogue computers (see page 238). The third type are symbolic models that use abstract symbols and mathematical relationships to represent the situation. It is this last type that digital computers use in simulation.

There are four main situations in which we might choose to solve a problem by simulation on a computer. The first is where the situation might be too dangerous to experiment with — for example, determining what level of radioactivity is safe in the countryside around a nuclear reactor.

The second situation, of which a good example is a model of the national economy, is where it would be almost impossible to find a pure mathematical solution to the set of equations that make up the problem. It is better to set up the equations in the form of a model on the computer and observe the effects on them of different actions and events.

The third case is where the problem to be analysed involves so much expense that any adjustments must be put into effect at the model stage before a commitment is made to the final version. In the planning of a proposed new airport for London, for example, extensive simulation work was done for the engineering and planning problems. This simulation investigated noise and other environmental considerations, as well as the flow of people and traffic around the proposed site.

Under Control
An important use of simulation is in education, where people are trained on models of real systems. An example is Atari's simulation of a nuclear power station, in which the user has to control the various cooling systems to prevent the reactor from overheating. The documentation explains fully the various control functions within such a power station, and there is also a demonstration mode that will run the program for you

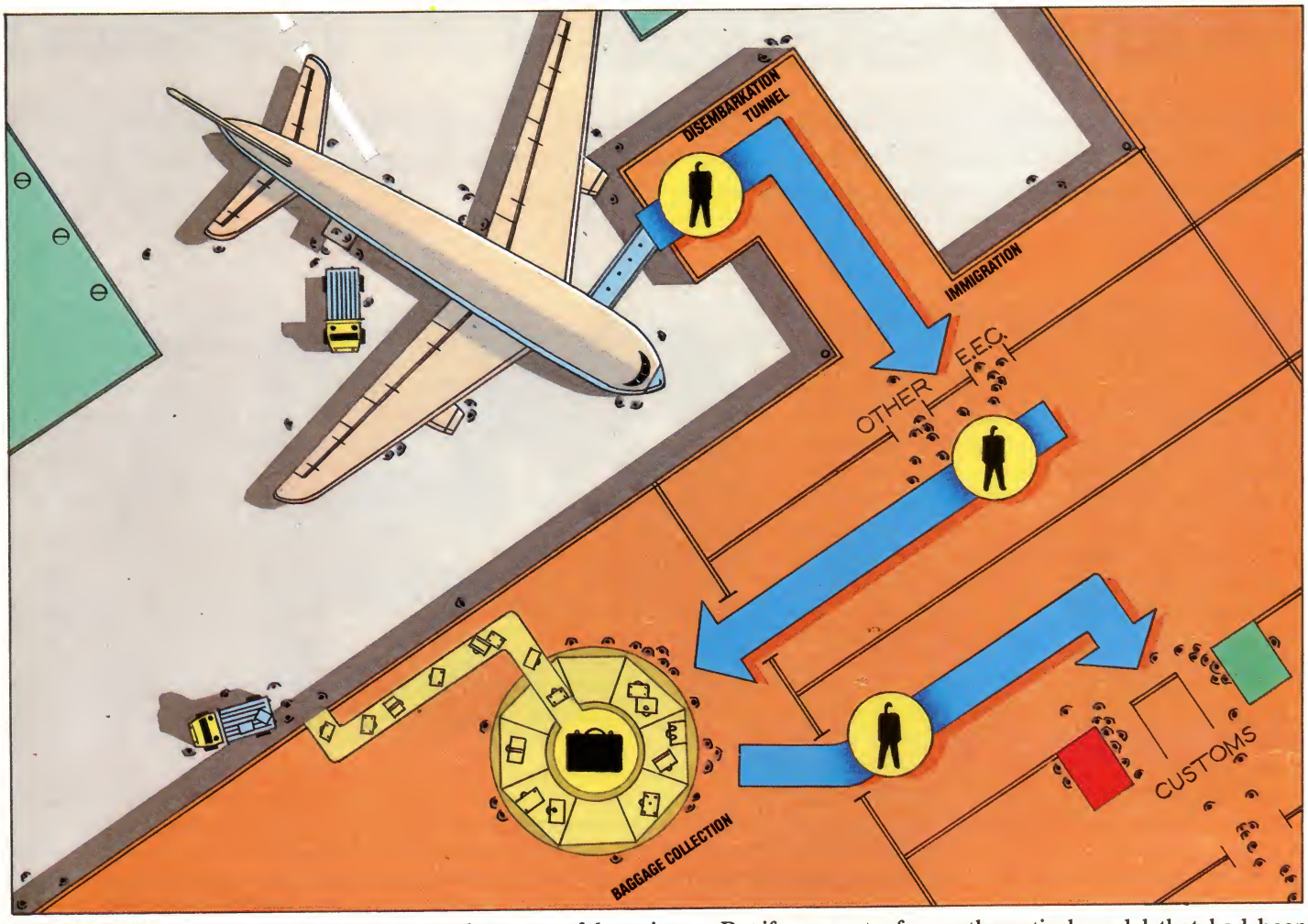
SOFTWARE COURTESY OF PILOT SOFTWARE
IAN McKINNELL

The other situation in which a model is valuable is one involving a problem that is totally theoretical and in which it is impossible to carry out physical experiments. Astrophysicists, for example, speculate on how stars are formed, and models are used to evaluate one cosmological theory — say the 'Big Bang' theory — against another.

In every simulation the first job is to construct the model. A model is made by studying the situation and deciding what the important elements are and how they interrelate.

Systems and their models fall into two families: 'deterministic' or closed systems, and 'stochastic' or open-ended systems. When a family budgets its expenditure the money can be allocated in many ways, but the books must balance in the end — making it a deterministic system. However, if the family based every decision to spend money on the toss of a coin, rather than the amount of money left in its account, the system would be stochastic.

With theoretical simulations it is not possible to be absolutely certain that your chosen model is the correct one. People thought that the earth was the centre of the universe until Copernicus provided a far simpler mathematical model, with the sun at the centre. Today astronomers observing distant galaxies of stars find that they are all receding from us at ever-increasing speeds, which once again



ROY INGRAM

Jumbo-Sized Problem
Computer simulation was extensively used in the design of the new Terminal 4 at Heathrow. The program first had to simulate the pattern of aircraft arriving throughout the day, with varying numbers of passengers and suitcases. It then simulated the 'processes' through which the passengers must pass. The model checked that the system could cope with the expected volume of traffic, and suggested optimal planning measurements

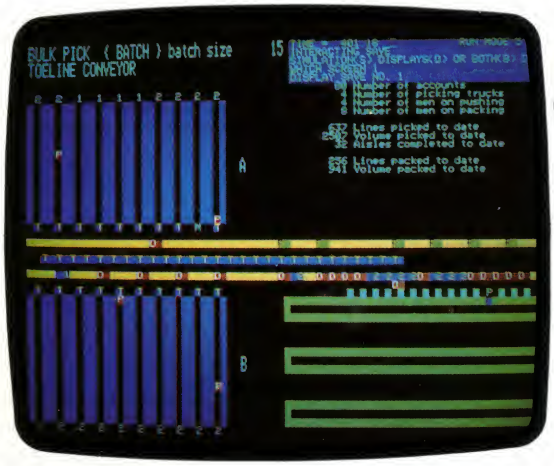
suggests we are at the centre of the universe. But if we adopted an earth-centred model of the universe, we should be misled — as an alternative model shows. If you splatter ink spots on a rubber balloon to represent stars and then inflate it, every ink spot will recede from all the others, yet none is at the centre of the balloon.

However, there are many advantages in using models. They help to formulate better theories, they speed up analysis, they allow modifications to be tried and, most importantly, they are cheaper than the real thing. Their use also allows for creative leaps of theory. The laser, for example, was invented by someone pursuing an aspect of a mathematical model that had been previously overlooked.

BL Systems, a subsidiary of British Leyland, market a multi-purpose simulation system that was originally developed for the design of production lines and automated warehouses at Cowley and Longbridge. The system is called 'See Why' and uses graphic displays to show results rather than the traditional lists of statistics. The system has sold well and has been used by the British Airports Authority to model the new Terminal 4 at Heathrow.

A typical problem that a simulation system can deal with is queuing. At an airport, for example, if the wind suddenly changes and only one of the landing strips is available, the aeroplanes will have to queue. The planes have only a limited reserve of fuel on board and it takes a specific length of time to land each plane. Rules of queuing will be programmed into the system. In these simulations random number generators (see page 209) are used to create unexpected events, such as random arrivals of aircraft.

Simulation is an important application area for digital computers and has even spawned new languages that have been specially written for simulation projects (for example, GASP, SIMSCRIPS and GPSS). As the world grows more complicated, the use of models to simulate problems will become more important.



COURTESY OF BL SYSTEMS

See Why
BL Systems, a division of British Leyland, developed a microcomputer modelling package for designing their new production lines and plants. The package, called 'See Why', has since been sold commercially. Although the emphasis of the program is on modelling processes, it does feature graphical output in the form of schematic diagrams. Numbers by the side of each stage in the process indicate how the work is progressing, and highlight any problems

Dotting The I's

The 'character generator' is the section of your computer's memory that defines the way characters look on the screen. On some systems it is possible to design your own symbols

We have already seen in the Basic Programming course (see page 214) that all alphanumeric characters — and graphic symbols if your computer has them — are stored in RAM memory in the form of eight-bit codes (usually ASCII), so that one character occupies one byte.

When information is printed on the screen, the codes for each character are put into a reserved area of memory called the 'video RAM'. If, for example, the letter A is printed in the top left-hand corner of the screen, the first byte of video RAM will contain the code 65 (ASCII for A). If a C is printed below the A, and the computer has a 40-column screen, then the value 67 will be found in the 41st location of video RAM, and so on. How does the computer convert the value 65 into the pattern of dots that makes up the character A on the screen? The answer lies in a device called a 'character generator'.

A character generator is simply a collection of patterns stored as bits in memory. Home computers have the character generator stored in ROM — thus permitting an immediate display of characters when the machine is switched on. The character generator may be incorporated into the ROMs that contain the BASIC interpreter and operating system, or it may be on its own ROM chip. Where the latter is the case, you will often find independent suppliers offering replacement ROMs that will produce a foreign character set, or a range of specialist symbols for, say, engineering or mathematics. However, an increasing number of machines permit the character generator to be transferred into RAM, which allows the programmer to design his own characters and symbols.

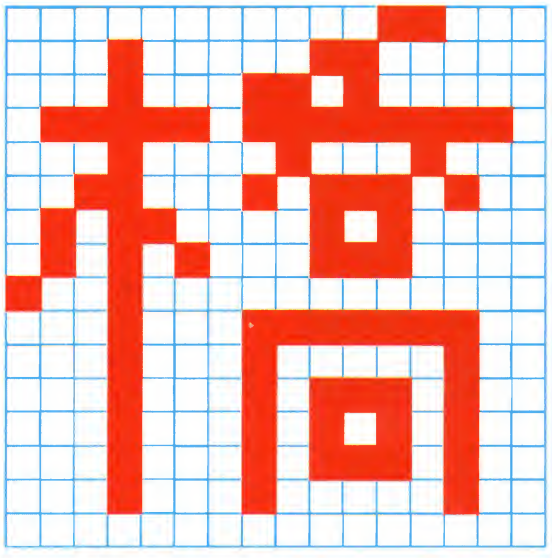
All characters are constructed on a matrix of dots, which on most home computers is eight by eight, although a larger matrix would result in greater legibility and a wider range of displayable characters. Characters are designed by filling in squares on the grid. They are defined by representing each filled square with a 1, and blank squares with a 0, giving a total of 64 bits, or eight bytes for each character.

The first byte in the character generator would represent the bit pattern for the top line of the first character in the table. If the computer can display only the ASCII characters with codes from 0 to 127, then the character generator will require 128×8 bytes (1 Kbyte of memory).

The difficulty for the computer is that when the raster scan on the television screen is generating the topmost line of the display, it needs to produce the topmost line of dots for the character positioned at the top left of the screen, followed by the topmost line of the character positioned to the right of it, and so on across the screen. Then, when the raster scan starts on its second pass, it must find and display the second line for each of the characters on the top row of the screen.

The video circuitry achieves this by having two independent counters. One counter keeps track of which location in the video memory corresponds to the point that the raster scan is passing over. The other counts the raster scan lines, starting at zero for the first line and reaching seven for the eighth, and then resetting to zero on the ninth, and so on. Thus, the computer looks up the ASCII or display code from the video memory, multiplies it by eight and adds on the current value of the line counter. This gives it an address in the character generator for the eight-bit pattern that corresponds to the correct row of the character it is currently scanning.

Let's take the example of generating the character A, which has an ASCII code of 65. We can calculate that the first line of the character will be stored in byte number 520 (65×8+0); the second line in byte 521 (65×8+1); the third line in byte 522 (65×8+2); and so on. All that remains is for the video circuitry to convert those eight bits into a sequence of voltages that will switch the scanning electron beam on and off to display the pattern on the screen.



Complex Characters
Japanese characters can be quite complicated, and the usual 8×8 matrix cannot display enough detail to make them readable. The character for 'bridge', shown here, is just readable on a 16×16 matrix. Better clarity is obtained by using a 24×24 dot matrix

Tandy Color Computer

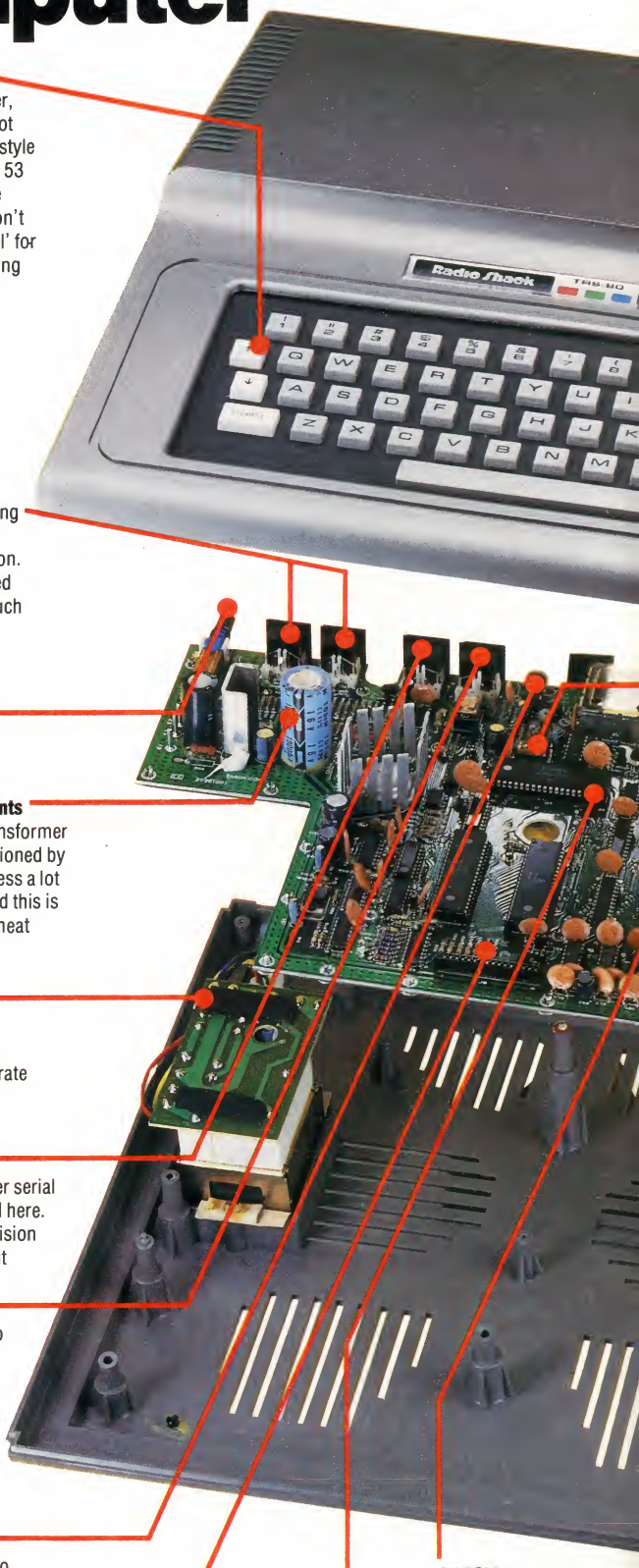
Very similar in design to the Dragon 32 (though the Tandy came first), this computer is well supported by peripherals — from digitisers to ink-jet printers

Despite being quite different in appearance, the Tandy Color Computer bears a remarkable resemblance to the Dragon 32 (see page 130) in the way it operates. They are so compatible, in fact, that many programs that run on one will run on the other.

There are other similarities, as well. Both have the same type of cartridge port and the same CPU — a 6809E. This is a very powerful processing unit that has been used in few other home computers. But in some respects, this CPU is ill-matched with the other components of the machine. Normally, the 6809 is used in machines designed for professional use. Its power is rather wasted in the Tandy Color Computer, however, which has a relatively low clock speed of 895 KHz.

The machine comes with its own version of BASIC — Tandy Color BASIC. Despite the low clock speed, this is acceptably fast in operation, thanks to the sophistication of the CPU. It has several specialised command words that are designed to make the hardware easier to use. One of the Tandy Color Computer's features is the 6847 video controller, a programmable device that generally produces a screen format of 16 lines of 32 characters. Normally, the display is of black

Tandy Graphics Tablet
The GT-116 graphics tablet can be used either as a digitiser for tracing drawings and images into the computer, or as a pointing device for selecting items from menus displayed on the computer screen. The tablet takes A4-size paper, and costs £300



The Tandy Keyboard
The Tandy Color Computer, unlike the Dragon, does not have a normal typewriter-style keyboard, but instead has 53 square buttons. These are adequately spaced, but don't have quite the correct 'feel' for long periods of touch-typing

Joystick Connectors
Fully proportional and including a signalling line, which is generally connected to a button. These interfaces could be used for other analogue signals, such as those from laboratory experiments

On/Off Switch

Power Supply Components
The output from the transformer is smoothed and conditioned by these parts. In the process a lot of heat is generated, and this is dissipated by the large heat sinks

Transformer
The in-built power supply makes for a tidier setup, with fewer trailing wires and separate boxes

RS232 Serial Port
Printers, modems or other serial devices can be connected here. The machine has no provision for parallel input or output

Tape Interface
Remote motor control is also provided on this interface

Channel Selection
The TV output can be tuned to two different channels. A selection is made by changing the position of this switch

8K ROM
Contains Tandy Color BAS and low-level routines

6847 Video Controller
The style and colour of the screen display is controlled by programming this chip, either from BASIC or in machine language

Keyboard Connector
A laminated plastic ribbon cable joins the keyboard to the computer through this port

letters on a green background, though either foreground or background may be set to any one of nine colours, and the format can be altered.

Facilities for sound generation are provided, but with only one channel and no envelope control or 'white noise' generation this machine is less flexible than others.

The Tandy Color Computer has the same type of ROM cartridge slot as the Dragon, and a wide range of software is available in this format. There is the usual variety of games cartridges: chess, cards, draughts, oil-prospecting, and pinball. The available software includes music, arithmetic and typing cartridges. There is also a good range of utility programs for budgeting, filing and letter writing, as well as some program development aids for BASIC or machine code. In addition there is a very handy diagnostic ROM, designed to ensure that the machine is working correctly.

An RS232 serial interface at the back of the machine allows for the use of a range of external devices, including serial printers. This port can also be used for communication with other computers using a modem (see page 216). With such a device and the videotex cartridge, various databases can be searched, or electronic mail sent.

Two joystick connectors are provided on the back of the machine, each with two axes of movement and a fire button. This seems to present a small problem in Tandy Color BASIC: a stream of characters is generated when the button is pressed, but Tandy say this can be ignored.

The joysticks are much better than many others, with variable movement instead of the usual simple push-pull type of mechanism, and could be used to read other analogue (variable value) inputs, such as those from physics experiments.

Expansion of the machine is through the cartridge port, and up to four $5\frac{1}{4}$ inch disk drives can be connected — giving a respectable total store of 626 Kbytes.

TANDY COLOR COMPUTER

PRICE
£99.95 standard with 16K RAM, £139.95 with extended BASIC

SIZE
369 × 344 × 94mm

CLOCK SPEED
0.895MHz

MEMORY
4 Kbytes of RAM; 8 Kbytes of ROM, expandable to 32 Kbytes

VIDEO DISPLAY
16 lines of 32 characters. 9 colours with independent background and foreground setting. 127 pre-defined characters and 127 user-definable characters

INTERFACES
RS232 serial, cassette, two proportional joysticks, TV interface on two channels

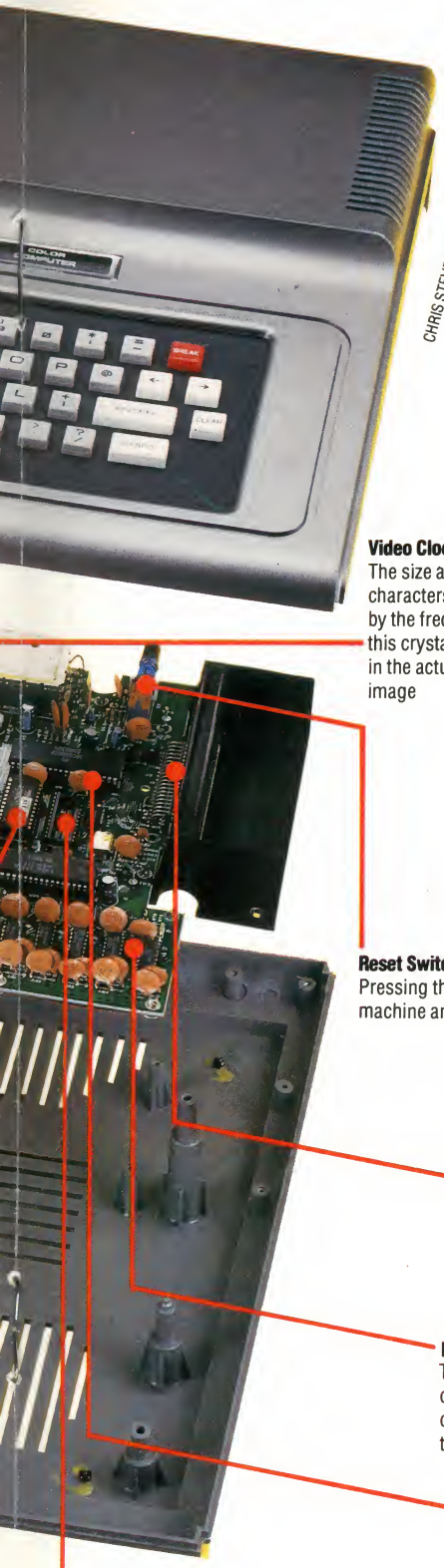
LANGUAGES SUPPLIED
BASIC

OTHER LANGUAGES AVAILABLE
6809 machine code with Assembler packages

COMES WITH
Installation and BASIC manuals, TV lead

KEYBOARD
53 individual keys, with typewriter spacing

DOCUMENTATION
The user's manual is uneven in presentation, but is very comprehensive and a perfectly adequate guide to the machine



CHRIS STEVENS

Video Clock
The size and number of characters on the screen is set by the frequency generated by this crystal, which is also used in the actual production of the image

Reset Switch
Pressing this will clear the machine and restart it

Cartridge Slot
Used to connect disk drives and operating software as well as ordinary game and program cartridges

RAM
This machine has 16K of dynamic RAM in 8 chips, each chip comprising one bit from the 8-bit byte

6809E CPU
An advanced chip with internal 16-bit arithmetic, operating on an 8-bit databus, it can address 64K of memory

Spare 8K ROM Socket
Could be used for additional utilities or extensions to BASIC

Tandy Printers
The Tandy Color Computer is well supported by a range of printers marketed by Tandy themselves. The CGP-220, for example, uses the ink-jet principle to produce seven colours, and is silent in operation. The cost is just under £500. At the other end of the scale, the CGP-115 at £150 uses four miniature ballpoint pens to produce colour printing and graph plotting

CHRIS STEVENS

New Entries

In order to insert a new entry in an array, it is first necessary to find a blank space. The binary search is an efficient way of achieving this

We saw in the previous instalment how a file of data is made up of records, which are divided into fields, each of which can be given access to the other fields through an indexing field. Now we will consider some of the techniques of searching through these lists.

Creating records for our address book is not difficult. Assume that a separate string array exists for each of the fields in the record. These can be called FLNAME$ (for full name), STREET$, TOWN$ and PHONE$ (we'll talk later about our use here of a string variable rather than a numeric variable for the phone number field). In the list of eight desirable functions of the address book program, number six was the facility to add new entries. If these eight choices were presented on the screen at the beginning when the program was run, selecting 6 would take you to an input routine of the type presented as an exercise.

Assume there are already a number of entries in the address book, but you can't remember how many. It is essential that new entries are not written over existing entries, so one of the tasks of the program might be to search through the elements in one of the arrays to find the first one containing no data.

Searching through an array to see whether an element is 'occupied' is not difficult. String variables can be compared in BASIC just as numeric variables can. IF A$="HOME" THEN . . . is just as valid as IF A=61 THEN . . ., at least in most versions of BASIC. If any of the arrays in our address book has an entry already, this will consist of at least one alphanumeric character. An 'empty' element will contain no alphanumeric characters, so all we need to do is search through the elements, starting at the beginning, until we find one containing no characters.

If there are arrays for the name, the street, the town and the phone number, we will have four arrays with one element in each for each field in the record. Since all these fields 'go together', the 15th record will have its name data in the 15th element of the name array, its street data in the 15th element of the street array, the town data in the 15th element of the town array, and the phone number data in the 15th element of the phone number array. We therefore need only to search through one of these arrays to find an empty element; we don't need to check all the arrays.

If the variable POSITION represents the number of the first free element in any one of the arrays, a program to locate POSITION (assuming it is not already known) could be as simple as this:

```
PROCEDURE (find free element)
BEGIN
  LOOP
    REPEAT UNTIL free element is located
    READ Array (POSITION)
    POSITION = POSITION + 1
    IF Array(POSITION) = " "
      THEN note POSITION
      ELSE do nothing
    ENDIF
  ENDLOOP
END
```

In BASIC, this could be as simple as:

```
1000 FOR L = 0 TO 1 STEP 0
1010 LET POSITION = POSITION + 1
1020 IF FLNAME$(POSITION) = " " THEN LET
   L = X
1030 NEXT L
1040 REM rest of program
```

Note that the value of X in line 1020 is the value required to terminate the FOR . . . NEXT loop and this value varies from machine to machine (see Basic Flavours). It is also important to note that this is a program fragment, and it is assumed that FLNAME$() is DIMensioned and that POSITION has been initialised. To run this fragment as a program on its own, you must DIMension FLNAME$() and initialise POSITION and X, at some point before line 1000.

Although we have used the FOR X = 0 TO 1 STEP 0 technique before, this is a good place to examine in more detail how it works. Usually, a FOR . . . NEXT loop in BASIC 'knows' beforehand how many times the program fragment is expected to repeat. If you want to repeat something 30 times, FOR X = 1 TO 30 will do admirably. This time, however, we are simulating a REPEAT . . . UNTIL loop. Although ordinary versions of BASIC do not have REPEAT . . . UNTIL on offer, it is easy enough to simulate using FOR . . . NEXT. As long as the test in line 1020 fails, L (the FOR . . . NEXT loop counter) remains at the value 0, with 0 added to it at every iteration (repetition of the loop); while line 1010 causes POSITION to be increased by 1 every iteration. When the test in line 1020 is true (that is, when an empty element of FLNAME$() is found), L is set to the value X, and the FOR . . . NEXT loop is terminated at line 1030. This leaves POSITION pointing to the first free element of FLNAME$().

POSITION is a value we are likely to need to

establish early, each time the address book program is used, and one that is likely to need updating several times during the use of the program. It is therefore going to be one of our 'global' variables, and establishing its value will need to be part of an 'initialisation' routine. This can be done every time the program is run, or a 'flag' can be created which indicates whether or not the value of POSITION has changed since the program was last run. The latter approach is not difficult, but at this stage it creates an unnecessary complication. We'll keep things simple and find the value of POSITION as one of the early tasks whenever the program is run.

Let's revise the activities we want the computerised address book to do and see if we can move towards an overall program strategy. This time we'll be slightly more rigorous and assume that each of the activities will be dealt with as a separate subroutine (the name of which will be indicated by being enclosed in asterisks).

1. Find record (from name)	*FINDREC*
2. Find names (from incomplete name)	*FINDNAMES*
3. Find record (from town)	*FINDNMTWN*
4. Find records (from initial)	*FINDINIT*
5. List records (all)	*LISTRECS*
6. Add record	*ADDREC*
7. Change record	*MODREC*
8. Delete record	*DELREC*
9. Exit program (save)	*EXPROG*

We now know, in broad terms, what the desired 'inputs' and 'outputs' of the program are, so we can already start thinking in terms of a main program. All the detailing can be done through the process of top-down programming and coded in the various subroutines. We know that several things will need to be initialised, including the value of POSITION. We know that, as the program is to be menu-driven, we will be presented with a set of choices whenever the program is run. We also know that, whatever our response to the choices presented, we will want one of them at least to be executed.

So the body of the main program can already take shape:

```
MAIN PROGRAM
BEGIN
   INITIALISE (procedure)
   GREET (procedure)
   CHOOSE (procedure)
   EXECUTE (procedure)
END
```

In BASIC it would look like this (with line numbers substituted for the subroutine names):

```
10 REM H.C.C. ADDRESS BOOK PROGRAM
20 GOSUB *INITIALISE*
30 GOSUB *GREET*
40 GOSUB *CHOOSE*
50 GOSUB *EXECUTE*
60 END
```

The *GREET* subroutine or procedure would display a greeting on the screen for a few seconds, followed by the menu. The greeting could, perhaps, look like this:

```
*WELCOME TO THE*
*HOME COMPUTER COURSE*
*COMPUTERISED ADDRESS BOOK*
(PRESS THE SPACE-BAR WHEN READY TO CONTINUE)
```

In response to the request to press the space bar, the program will branch to the *CHOOSE* subroutine and the user will be presented with a screen like this:

```
*DO YOU WISH TO*
1. FIND A RECORD (from a name)
2. FIND NAMES (from part of a name)
3. FIND RECORDS (from a town)
4. FIND RECORDS (from an initial)
5. LIST ALL RECORDS
6. ADD A RECORD
7. CHANGE A RECORD
8. DELETE A RECORD
9. EXIT AND SAVE
*CHOOSE 1 TO 9*
*FOLLOWED BY RETURN*
```

At this point, the program will branch to the appropriate subroutine depending on the number entered. The structure of the program is now beginning to take shape. All options except number 9 (to EXIT and SAVE) will need to end with an instruction to return to the *CHOOSE* subroutine. But there are many details of the internal organisation of the data that we have not considered. We will come to these later.

Let's assume that we are running the program, that it already has all the records in it that we need, and that we want to search for a full record by inputting a name only. This calls for option 1 — FIND A RECORD (*FINDREC*). Before we attempt to design this part of the program, let's consider some of the problems involved in computerised search routines.

Searching

Textbooks on programming techniques tend to deal with searching and sorting together. Readers may recall that we have already touched on sorting in a program designed to sort names into alphabetical order (see page 134). Both sorting and searching raise interesting points about how data is organised — in a computer or any other information system.

If a 'manual' address book comprised a note-book without a thumb index, and if entries were added when new names and addresses were thought of, without being sorted into alphabetical order, we would have a data structure known as a 'pile'. A pile is a set of data collected in the order in which it arrives. It is obvious that a pile is the least effective way of organising data. Every time you want to find someone's address and telephone number you have to look through the whole address book. The same is usually true of

computer systems, though there are occasions when the criteria by which data is accessed are so unpredictable that a pile is as good a data structure as any.

A more organised data structure, and one much easier for both people and computers to use, is achieved when the data is organised according to a recognised and simple system. A telephone directory is a good example of a set of information (names, addresses and telephone numbers) where the name field is ordered according to simple rules of alphabetic sequencing. The numbers themselves are, to all intents and purposes, randomly ordered, but the names — which are more 'meaningful' — are organised according to easy-to-follow rules.

Inasmuch as we have thought about the internal organisation of the data in our computerised address book, the data is organised as a pile, with one record being stored in name array element X, street array element X and so on, and the next record being stored in name array element X+1, street array element X+1 and so on. Finding a particular item of data — BILL SMITH, for example — would therefore involve looking at the first element in the name array and seeing if it was BILL SMITH, looking at the second element and seeing if it was BILL SMITH and so on until we had either located the field or discovered that there was no entry for BILL SMITH.

If the data we want to search for has already been ordered into a recognisable structure, we can see how it will simplify the search. Suppose you have a database on football teams, and one of the fields in the records is the score for a particular week. A powerful database might allow you to find which team or teams had scored 11 goals in that week. Here is the array holding team scores for the week in question:

1,6,2,2,1,9,0,0,2,1,4,11,4,2,12,5,2,1,0,1

It should be obvious that the scores are in team order and not in score order. Twenty teams are involved and only one team actually managed to score 11 goals that week. This was the 12th team entered in the array. With unstructured data like this, the only way to find the information you want is to look at the first element and see whether it was 11; if it was not, look at the next element to see whether it was 11, and so on until either an 11 was located or no element equal in value to 11 was found.

If we analyse this data, we will see that there was a total of 20 scores, ranging in value from 0 to 12. This example is relatively trivial, and even if we had to search through every item it would not take long to discover that 11 was in the 12th element of the array. But what if there were thousands of elements in a large array? Searching through numerous unstructured data items could slow down a program to an undesirable extent.

The solution is to order the data first, so that searches can take place far more quickly. Here is the array of scores again, arranged in numerical order:

0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12

If we know the number of teams is 20, then the quickest way to find the position of the score we want is to split the array into two parts and search only the part likely to contain the number we want. Remember that sifting through large quantities of data is likely to take far more time than simple arithmetic operations such as dividing a number by two. The algorithm for locating the score would now look like this:

```
Find the array containing the scores
Read the number we want to search for
Find the length of the array
Find the midpoint of the array
Loop until the number is located
   If the item at the midpoint is equal to the
   number we are searching for, then the
   number has been located
   If not, see if the number sought is larger or
   smaller than the number at the midpoint
   If the number sought is larger than the
   number at the midpoint, then find the
   midpoint of the upper part of the array
   If the required number is smaller than the
   number at the midpoint, then find the
   midpoint of the lower part of the array
   (Repeat this until the number is located)
```

This can be formalised to:

```
BEGIN
   Find the array of scores
   INPUT NUMBER (to be searched for)
   LOOP until the number is located
      IF NUMBER = (midpoint)
         THEN note position of midpoint
         ELSE
            IF NUMBER > (midpoint)
               THEN find midpoint of upper half
               ELSE find midpoint of lower half
            ENDIF
      ENDIF
   ENDLOOP
   IF NUMBER is located
      THEN PRINT position of midpoint
      ELSE PRINT "NUMBER NOT FOUND"
   ENDIF
END
```

If you think through this program in pseudo-language you will see that it cannot fail eventually to locate the number being searched for if it exists in the array. Let's develop this pseudo-language until we can arrive at a working program. This process of searching by repeated subdivision is called a 'binary search'.

A program in BASIC based on the pseudo-language above is presented for you to try. It creates an array and reads in the scores from a data statement. It then prompts for the score to be searched. If it finds the score, it prints the element of the array the number was found in.

```
10 REM A PROGRAM TO LOCATE A NUMBER IN AN
   ARRAY
20 DIM SCORES(20)
30 FOR Z = 1 TO 20
40 READ SCORES(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT"INPUT SCORE ";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP — BTM
130 LET MD = BTM + INT(L/2)
140 IF N = SCORES(MD) THEN LET Z = X
150 IF N > SCORES(MD) THEN LET BTM = MD
160 IF N < SCORES(MD) THEN LET TP = MD
170 NEXT Z
180 PRINT "THE SCORE WAS IN ELEMENT NO.
   ";MD
190 END
```

Again, note that X will need to be initialised according to your machine's requirements (see Basic Flavours).

If the data held in a file or array is fairly regular, as in the case of a telephone directory, where names are distributed reasonably evenly across the alphabet, then the binary search is an efficient way of finding a particular entry. However, it is by no means the most efficient, and there are alternative algorithms that can find the data using fewer iterations. One such is the technique of 'hashing', where the program makes an educated guess at the approximate location of the entry, refining the guess until it is found. Such methods, however, are beyond the scope of this course, and the binary search method is sufficient for our needs.

Exercises

If you run this program, you will see that it works provided you enter a score that exists in the array. If you enter a score such as 3, which is not in the array, the program fails to terminate and no error message appears. If you type in 12, which exists in the array, the program fails to locate it. The program also assumes that every number in the sorted array will be different but, as you can see from the data statement, several numbers occur more than once. The program neither detects this nor reports all the locations where the number occurs.

Your task is to:

1. Analyse the program and find out why it cannot locate a score of 12
2. Modify one line of the program to rectify this defect
3. Establish why the program is unable to handle numbers that do not exist in the string and devise a strategy to overcome this defect.

On page 235 of The Home Computer Course we featured a number of revision exercises to help you assess your progress in the Basic Programming course. See page 280 for the solutions.

Basic Flavours

SPECTRUM

The following is the Spectrum listing for the first BASIC program fragment:

```
100 DIM F$(6,4)
110 LET POSITION = 0
120 LET F$(1) = "MIKE"
130 LET F$(2) = "KATE"
140 LET F$(4) = "MARY"
1000 FOR L = 0 TO 1 STEP 0
1010 LET POSITION = POSITION + 1
1020 IF F$(POSITION) = "  " THEN
       LET L=2
1030 NEXT L
1040 PRINT "NO. OF 1st FREE E'LEMENT
   IS ";POSITION
1050 STOP
```

Notice that lines 100 to 140, as well as line 1040, transform the program fragment (lines 1000 to 1030) into a working demonstration program. The values and format of these extra lines can be changed to investigate the working of the program fragment.

The second program for the Spectrum is:

```
10 REM A PROGRAM TO LOCATE A
   NUMBER IN AN ARRAY
20 DIM S(20)
30 FOR Z = 1 TO 20
40 READ S(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,
   9,11,12
70 LET L = 20
80 LET BTM = 1
90 LET TP = L
100 INPUT"INPUT SCORE";N
110 FOR Z = 0 TO 1 STEP 0
120 LET L = TP—BTM
130 LET MD = BTM + INT(L/2)
140 IF N = S(MD) THEN LET Z = 2
150 IF N>S(MD) THEN LET BTM = MD
160 IF N<S(MD) THEN LET TP = MD
170 NEXT Z
180 PRINT"THE SCORE WAS IN ELEMENT
   NO. ";MD
190 STOP
```

VARIABLES

For variations in variable names, see previous 'Basic Flavours' (page 257).

STEP 0

In both programs in the main text there is a reference to ". . . THEN LET Z = X". The values for the variable X are:

Oric-1	replace X by 1
Dragon 32	replace X by 1
Lynx	replace X by 2
BBC Micro	replace X by 2
Commodore 64 and Vic-20	replace X by 1

LYNX

Both programs in the main text will run on the BBC, the Dragon 32, the Lynx, the Oric-1, the Commodore 64 and Vic-20 provided that the Basic Flavours concerning variable names and Step 0 are implemented. The Spectrum listings differ from the norm in the DIM statement in line 100 above, and in the test in line 1020 above, otherwise they may be used as a guide to implementation on the other machines. The Lynx version of line 100 above is:

```
100 DIM F$(4)(6)
```

TONY LODGE

Sound Advice

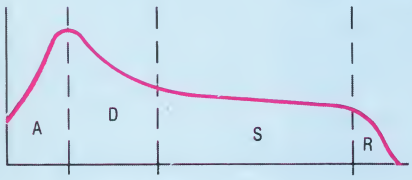
Continuing our explanation of computer music jargon

As part of our series on generating sound on a microcomputer we now look at some of the most advanced features on home computers.

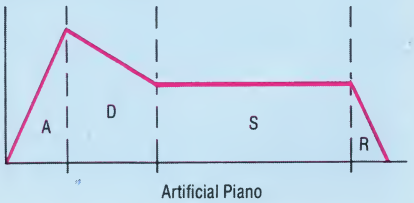
Envelope Generators

The envelope of a sound is the pattern of its volume changes, from the instant it is played until it dies away. Envelopes similar to those of a piano note and other instruments are shown in the diagram. Envelopes are commonly divided into four parts, usually called Attack, Decay, Sustain and Release (ADSR). In the case of the piano, volume rises rapidly to its highest level after the piano key is pressed (attack), then settles down more slowly (decay) to the roughly constant volume (sustain level) while the key is held down, and finally falls rapidly to zero (release) when the key is let go. Notice that sustain is a volume level, whereas attack, decay and release are intervals of time. A device that can control all four aspects of an envelope is called an ADSR generator.

Any device that can turn a sound on and off is an envelope generator of sorts.



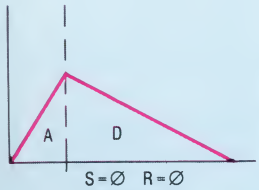
Natural Piano



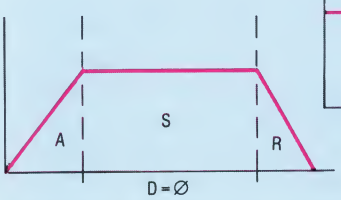
Artificial Piano

Most computers have nothing more sophisticated than a 'beep' generator, which is just a switch to turn on the sound at a constant volume for a specific time. In this case the A, D and R elements are equal to zero.

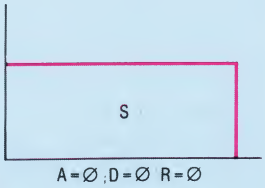
The Oric-1 provides some envelope control, but the most versatile home computers in this respect are the BBC Micro and the Commodore 64, both of which have ADSR generators that can imitate the envelopes of most common instruments, as well as some 'unnatural' ones.



S=Ø R=Ø
Xylophone



D=Ø
Violin



A=Ø ;D=Ø R=Ø
Organ

...Light Reading

Simple animation using the POKE and PRINT commands

Having looked at the basic principles of computer graphics, we're now going to see how some simple animation can be achieved. First, however, it will be necessary to detail the ways in which characters can be made to appear on the screen and how their positioning can be controlled using a BASIC program. Two main methods are available to the programmer: the POKE command, and the PRINT command with its associated functions.

The POKE command places a number in any specified memory location. There is a special series of memory locations inside every computer, each relating to a specific character position on the screen. On a standard 25-row by 40-column screen, 1,000 locations are set aside for this purpose. Each location holds a number that corresponds to a particular character in that machine's character set. This may be the standard ASCII code of the character (see page 214) or a code designated by the machine's manufacturer. In addition to these character code locations there is usually another set of locations that hold information about the colour of a character displayed in any screen position

On the Commodore 64, for example, there are very few graphics commands in BASIC to help the programmer, and the POKE command is often used to create screen displays. The addresses of the locations that hold character codes run from 1024 to 2023, and the locations that hold the colour information have addresses from 55296 to 56295. For the Commodore, the character A has a screen code 1 and the colour black is represented by a colour code of 0; thus the commands required to place a black letter A in the top left-hand corner of the screen are:

```
10 POKE 1024,1
20 POKE 55296,0
30 END
```

Simple modifications can be made to display a row of black A's on the top line of the screen:

```
10 FOR X = 0 TO 39
20 POKE 1024+X,1
30 POKE 55296+X,0
40 NEXT X
50 END
```

The A's are produced by the FOR...NEXT loop, which increases the character and colour location addresses by one each time. If we incorporate a command to rub out an old A every time a new one is created then the A will appear to move across the

screen: a crude form of animation!

The Commodore character code for a space is 32. All that is necessary is for the program to place this in the appropriate location, one place behind where the new A is about to be displayed. Insert this line into the above program:

```
35 POKE 1024+X,32
```

Alternatively,

```
15 POKE 1024+(X- 1),32
```

Having to POKE locations with numbers to produce graphics is a laborious process. The method is probably best used to create static background displays using READ and DATA statements to enter the character codes before POKEing them to the correct location.

Most home computers have many graphics commands as part of the standard BASIC instruction set, allowing the user to create impressive and colourful displays using only a few simple statements. Commands to construct geometric patterns in high resolution are often included. With these instructions the user can plot points on the screen and connect them with straight lines; draw squares, arcs and circles; and colour the interiors of the shapes drawn.

It is often a straightforward matter to replot old shapes with the same colour as the background, which has the effect of rubbing them out. Rapid plotting, rubbing out and replotting at a new position is, in fact, the basis of simple graphic animation. The realism of the action largely depends on the speed at which the process can be achieved. Sprites are much more effective because they do not require unplotting as they move to a new position, and this greatly improves the speed at which they appear to move. Indeed, the development of sprites means that for the first time it is possible to write arcade-style games in BASIC, where previously machine code was essential.

The principles of moving graphics can be used in conjunction with simple BASIC programs. Many home computers have commands that allow the user to PRINT at specific positions on the screen, such as PRINT AT on the Spectrum and PRINT@ on the Dragon.

Star Formation

Here is a short program for the Dragon using PRINT. In this the variable X is the screen location where the star is to be printed. Notice that line 40 rubs out the old star as the new one is PRINTed.

```
10 CLS: REM CLEAR
SCREEN
20 FOR X = 160 TO 191
30 PRINT@X,"*";
40 PRINT@X-1," ";
50 NEXT X
60 END
```

Waveform

The waveform is the repetitive 'shape' of the signal produced by an oscillator (see page 247) and gives the sound its character. Two different instruments playing notes of the same pitch do not sound the same, and this is partly because the waveforms are different. The most common waveforms are square (or pulse), triangle and sawtooth — as shown in the panel.

Most home computers provide only one waveform, usually of the pulse type. This is why many have that unmistakable harsh synthetic sound.

The Commodore 64 is currently the most interesting computer musically, mainly because you can select any of the three basic waveforms on each of three oscillators. The waveforms can be modified by the use of filters, which alter the tone in much the same manner as the bass/treble controls on a hi-fi and have the effect of mellowing the sound. Even more useful is the ability to change these filter settings throughout the duration of a note. This enables you to simulate natural sounds more closely and produce more exciting unnatural sounds.

Noise

Noise is a complex type of sound made by random vibrations. The ear cannot pick out a repetitive pattern, so it does not hear any specific pitch. Imagine some everyday sounds like rain, wind and thunder. These noises do not sound the same because they are a combination of pure (unpredictable) noise with some dominant tones. Most microcomputers with a noise facility therefore allow you to modulate the noise in some way, or to mix it with pure notes. The possible effects range from 'whistling wind' to violent explosions.

Output

Output is usually through the television speaker. If this is the case you can connect the television set to your hi-fi via a video recorder. Some computers, however, can output sound only through a small built-in speaker. For these it is impossible to obtain good quality sound without modifying the hardware of the computer or buying an external add-on. Your computer may have an output suitable for connection directly to your hi-fi, making the effort involved in producing complex sound shapes worthwhile.

Crystal Clear

Liquid Crystal Displays, featured widely on watches and calculators, are now starting to appear on computers

Liquid Crystal Displays (LCDs) have been available since late 1973, when they first made their appearance in calculators. Later they were employed in the production of digital watches, and greatly contributed to the popularity of that type of watch. Now LCDs are beginning to find a place in the microcomputer industry. They are used in the A4-sized portables like Epson's HX-20 (see page 169), and the Tandy TRS80 Model 100 Portable Computer, as well as Sharp's high capacity PC 5000.

In order to understand what liquid crystals are, we must first appreciate that all matter varies its physical state depending on temperature and pressure, from solid (or crystalline) through liquid to gas. Only in the solid state is any regular alignment of a substance's molecules to be found. That is, except in the case of a very few substances where that regular alignment is maintained part way through into the liquid state. These substances, the nature of which is a closely guarded trade secret, are known as liquid crystals.

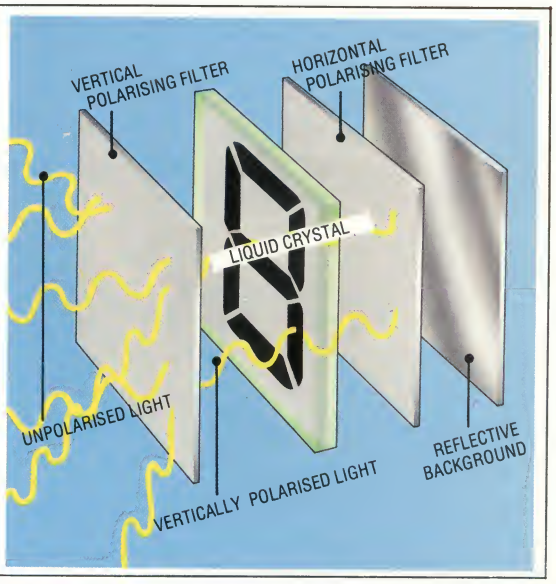
Until the mid-seventies, displays in calculators and watches were composed of bar-shaped Light Emitting Diodes (LEDs), arranged to form a rather angular version of a letter or number. But LED displays have several drawbacks: they require considerable amounts of power and are relatively large in size.

In the search for alternative methods of information display, it was discovered that the alignment of the molecules in liquid crystals could be altered by an electric current; and furthermore, this alteration was purely a local one. Once this principle had been established, it became possible to construct a medium for displaying information. The first step involved forming electrodes in the shape of a character on the inside faces of two sheets of glass. A very thin layer of liquid crystal was sandwiched between these, and a voltage applied. In ordinary light, nothing appeared to happen, but when polarising filters (see diagram) were applied to the back and front and the whole structure mounted against a reflective background, the desired effect was produced — a clearly-defined character against a neutral background.

The process by which this character is defined requires light to pass through the first filter, and in this way to be polarised vertically. It is then deflected through 90°, and thus is blocked out at the rear filter. In this way, the area of the liquid crystal to which a voltage has been applied

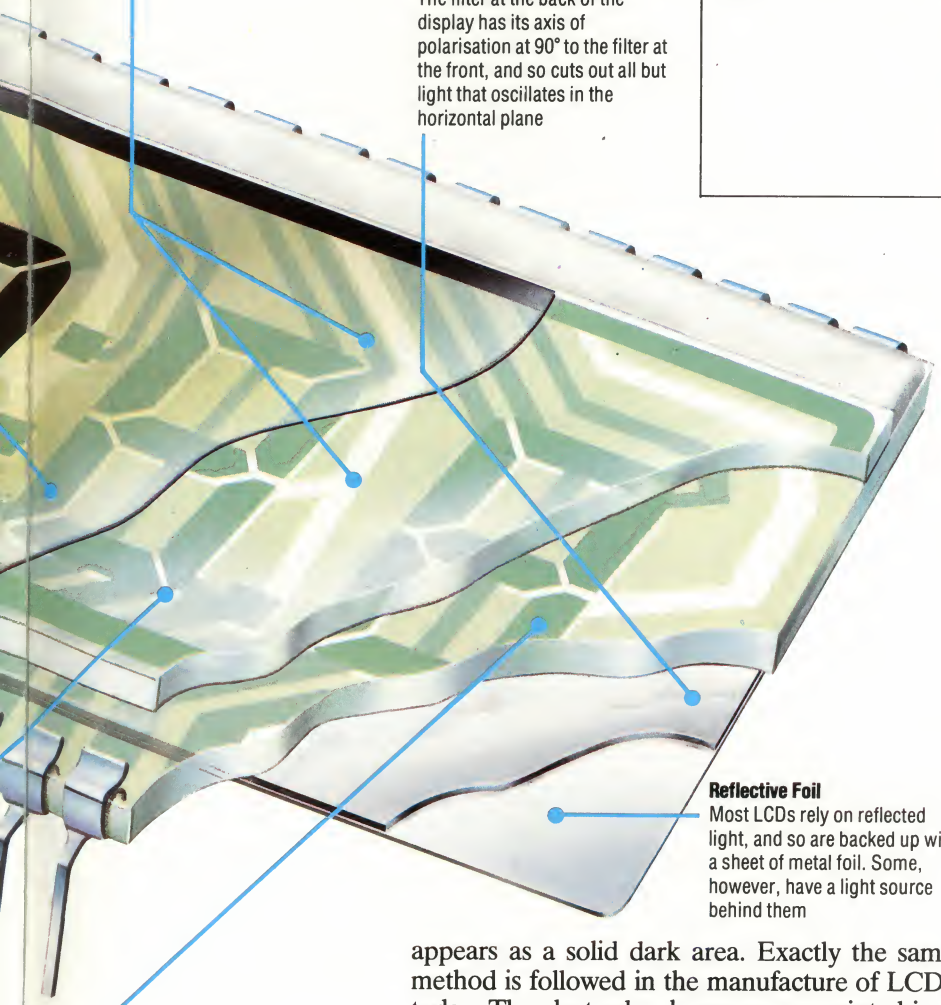
Angle Of View
One of the refinements of Epson's remarkably compact and sophisticated HX-20 portable computer is the angle of view adjuster. Liquid crystals are composed of long thin molecules that have magnetic poles positioned in the middle of their long sides. Applying an electrical voltage across their length causes them to try to twist end over end, while their natural stability attempts to keep them in place. The greater the current passed, the more they will twist

Polarisation
The filter at the front of an LCD only allows rays of light that have the electromagnetic oscillation vertically oriented to pass through it. The liquid crystal then turns the 'polarisation' through 90°, allowing it to pass through the rear (horizontal) filter and be reflected back. When a segment of the LCD is energised, it ceases to rotate the polarisation, resulting in a black image



Connection Paths
Each electrode is connected to its driving circuit by means of an almost invisible deposit of conducting ink on the surface of either the back or the front glass

Horizontal Polarising Filter
The filter at the back of the display has its axis of polarisation at 90° to the filter at the front, and so cuts out all but light that oscillates in the horizontal plane



Reflective Foil
Most LCDs rely on reflected light, and so are backed up with a sheet of metal foil. Some, however, have a light source behind them

Positive Electrodes
In this representation of the characters, each component is separate and individually addressable by the driver circuits

appears as a solid dark area. Exactly the same method is followed in the manufacture of LCDs today. The electrodes, however, are printed in a clear, colourless ink onto the surface of the glass and dry to near-invisibility.

Because of the requirement to produce a variety of characters from the same matrix, the original method — short bars combining to make angular characters — is still used, though dot matrix LCDs are becoming increasingly common. Because each dot in the matrix is individually addressable, the graphics capabilities are quite advanced: both continuous forms and conventional graphics characters can be produced.

It is theoretically possible to use dot matrix-addressed LCDs to act as the screen for broadcast television receivers, as well as monitors for use with computers. The main drawback to this is the lack of variability in the contrast. This was one of the problems explored during the development of the various flat screen televisions now becoming available. The size of the individual elements in the matrix needs to be reduced to something approaching that of a pixel on a cathode ray tube in order to achieve acceptable resolution. Another possibility would require the use of a number of LCDs sandwiched together and operating simultaneously. This method is currently used for the display of complex information where space is limited.

The response time of a high quality LCD at normal operating temperature (20°C/68°F) is about 70 milliseconds for the rise from neutral to black, and a further 80 milliseconds for the fall back to neutrality. This total of 150 milliseconds compares very unfavourably with the cathode ray tube's response of 0.00025 milliseconds. However, the LCD has a number of advantages. We have already discussed its compact size, but its low power consumption is perhaps the most significant advantage. A typical LCD consumes as little as 10 microwatts per square centimetre of the display.

There is one very interesting effect of varying the voltage passed through a liquid crystal. The tilt of the molecules is increased and decreased according to the voltage that is applied. Epson, for example, use this to very good effect in the HX-20 portable computer, by giving the display a viewing angle adjustment.

One further advantage is apparent in conditions of strong sunlight. The contrast of a cathode ray tube display is severely diminished in such circumstances, but because liquid crystals are perceived by the *absence* of light reflected from them, LCDs appear to become more contrasting and thus more readable as external light increases.

Although still only a decade old, LCD technology has already made significant inroads into markets traditionally reserved for cathode ray tubes. As the requirements of micro-miniaturisation become greater, it is expected that this technology will develop still further.

Answers To Exercises

How did you get on with our revision exercises on page 235? Here are some model solutions, though you may have found alternative methods that also work

On page 235 we posed nine problems, designed to test your skills at using the statements and functions commonly encountered in BASIC. Here are our suggested solutions.

If you've been following the Basic Programming course so far, you may have recognised the exercises as problems that have already been met and solved. Your solutions may be different from those we have suggested here. There is seldom only one way of solving a problem; your way may be as good as, or better than, ours.

If you find the solutions as puzzling as the questions, read pages 149 onwards again and study the solutions as you progress. If you did well on exercises 2, 4, 6 and 8 you have understood most of the lessons of these articles.

```
100 REM REVISION EXERCISE 1
200 INPUT "TYPE IN ANY NUMBER";A
300 INPUT "TYPE IN ANOTHER NUMBER";B
400 LET C = A + B
500 PRINT "THEIR SUM IS ";C
```

```
100 REM REVISION EXERCISE 2
200 LET A$ = "FIRSTWORD,"
300 LET B$ = "SECONDWORD"
400 LET C$ = A$ + B$
500 PRINT C$
```

```
100 REM REVISION EXERCISE 3
200 INPUT "TYPE IN ANY WORD";W$
300 LET L = LEN(W$)
400 PRINT "THE WORD YOU TYPED HAS ";L;"
    CHARACTERS"
```

```
100 REM REVISION EXERCISE 4
200 PRINT "HIT ANY KEY"
300 FOR C = 0 TO 1 STEP 0
400 LET A$ = INKEY$
500 IF A$ <>"" THEN LET C = 2
600 NEXT C
700 PRINT "ASCII VALUE OF ";A$;" IS ";ASC(A$)
```

See 'Basic Flavours', pages 175 and 215. On the Spectrum, replace line 700 with:

```
700 PRINT "ASCII VALUE OF ";A$;" IS ";CODE(A$)
```

```
100 REM REVISION EXERCISE 5
200 INPUT "TYPE IN ANY WORD";W$
300 LET L$ = RIGHT$(W$,1)
400 PRINT "THE LAST CHARACTER OF THE WORD
    IS    ";L$
```

See 'Basic Flavours', page 149. This exercise on the Spectrum reads:

```
100 REM REVISION EXERCISE 5
200 INPUT "TYPE IN ANY WORD";W$
250 LET N = LEN(W$)
300 LET L$ = W$(N)
400 PRINT "THE LAST CHARACTER OF THE WORD
    IS ";L$
```

```
100 REM REVISION EXERCISE 6
200 PRINT "TYPE IN A NAME IN THE FORM:"
300 PRINT "FIRSTNAME SECONDNAME"
400 PRINT "E.G. JILL THOMPSON"
500 INPUT "NAME ";N$
600 LET S = 0:LET L = LEN(N$)
700 FOR P = 1 TO L
800 IF MID$(N$,P,1) = "    " THEN LET S = P
900 NEXT P
950 PRINT "SPACE WAS THE ";S;"TH CHARACTER"
```

See 'Basic Flavours', page 149. On the Spectrum, replace line 800 above with:

```
800 IF N$(P) = "    " THEN LET S = P
```

```
100 REM REVISION EXERCISE 7
150 LET X$ = "TH"
200 PRINT "TYPE IN A NAME IN THE FORM:"
300 PRINT "FIRSTNAME SECONDNAME"
400 PRINT "E.G.JILL THOMPSON"
500 INPUT "NAME ";N$
600 LET S = 0:LET L = LEN(N$)
700 FOR P = 1 TO L
800 IF MID$(N$,P,1) = "    " THEN LET S = P
900 NEXT P
925 IF S = 2 THEN LET X$ = "ND"
950 PRINT "THE SPACE WAS THE ";S;X$;"
    CHARACTER"
```

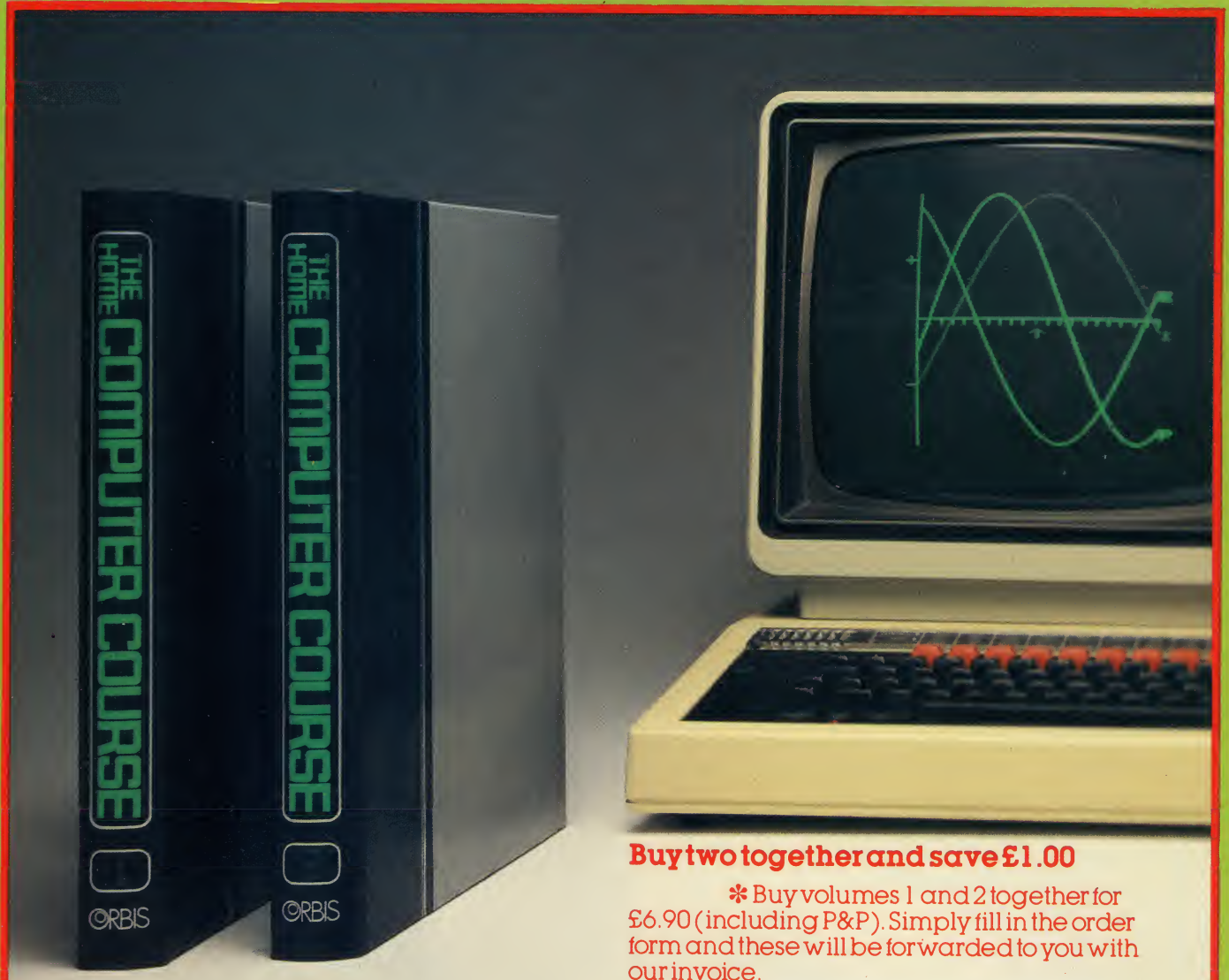
```
100 REM REVISION EXERCISE 8
200 INPUT "TYPE IN A SENTENCE ";S$
300 LET C = 1
400 FOR P = 1 TO LEN(S$)
500 IF MID$(S$,P,1) = "    " THEN LET C = C + 1
600 NEXT P
700 PRINT "THE SENTENCE HAS ";C;" WORDS"
```

See 'Basic Flavours', page 149. On the Spectrum, replace line 500 above with:

```
500 IF S$(P) = "    " THEN LET C = C + 1
```

```
100 REM REVISION EXERCISE 9
200 FOR C = 128 TO 255
300  PRINT "CHARACTER NO. ";C;" = ";CHR$(C)
400 REM A SHORT DELAY HERE
500 FOR D = 1 TO 500
600 NEXT D
700 REM END OF DELAY
800 NEXT C
```

THE HOME COMPUTER COURSE BINDER



Now that your collection of Home Computer Course is growing, it makes sound sense to take advantage of this opportunity to order the two specially designed Home Computer Course binders.

The binders have been commissioned to store all the issues in this 24 part series.

At the end of the course the two volume binder set will prove invaluable in converting your copies of this unique series into a permanent work of reference.

Buy two together and save £1.00

✻ Buy volumes 1 and 2 together for £6.90 (including P&P). Simply fill in the order form and these will be forwarded to you with our invoice.

✻ If you prefer to buy the binders separately please send us your cheque/postal order for £3.95 (including P&P). We will send you volume 1 only. Then you may order volume 2 in the same way – when it suits you!

Overseas readers: This binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in Issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain their binders **now**. For details please see inside the front cover.

Binders may be subject to import duty and/or local tax.

THE LAST WORD IN LOGIC

Mentathlete

Home computers. Do they send your brain to sleep – or keep your mind on its toes?

At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.

Provided, of course, it offers a whole battery of genuine mental challenges.

The Spectrum does just that.

Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!

The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.

And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.

No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.

For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.

sinclair